Claims

[c1] 1. In a database system, a method for performing recovery operations using an optimal number of recovery threads, the method comprising:
(a) spawning an initial recovery thread to perform recovery operations;
(b) measuring I/O (input/output) performance with the initial recovery thread;
(c) spawning a subsequent recovery thread to perform recovery operations;
(d) measuring I/O performance with the subsequent recovery thread; and
(e) as long as I/O performance does not degrade beyond a preselected percentage, repeating steps (c) and (d) for spawning a desired number of additional recovery threads.

[c2] 2. The method of claim 1, wherein I/O performance is measured over a given period of time.

[c3] 3. The method of claim 2, wherein the given period of time is about 1 second.

[c4] 4. The method of claim 1, wherein steps (c) and (d) are

repeated for spawning additional recovery threads, as long as I/O performance degrades by no more than about 15 percent.

[c5] 5. The method of claim 1, wherein steps (c) and (d) are repeated such that only a preconfigured maximum number of recovery threads may be generated.

[c6] 6. The method of claim 5, wherein the maximum number of recovery threads is limited to not exceed a count of databases that can be opened.

[c7] 7. The method of claim 5, wherein the maximum number of recovery threads is limited to not exceed one less than a count of database engines online.

[c8] 8. The method of claim 1, wherein step (e) further comprises:
when I/O performance measured for a just-spawned recovery thread degrades beyond the preselected percentage, putting the thread to sleep.

[c9] 9. The method of claim 8, further comprising:
after another recovery thread finishes, awaking the thread that has been put to sleep.

[c10] 10. The method of claim 1, wherein steps (c) and (d) are repeated up to a configured maximum number of

databases that can be recovered concurrently.

[c11] 11. The method of claim 1, wherein each recovery thread itself recovers a single database at a time.

[c12] 12. The method of claim 1, wherein a user of the system is able to specify a particular number of concurrent recovery threads, and wherein the system generates an advisory if the particular number of concurrent recovery threads specified is not optimal.

[c13] 13. A computer-readable medium having processor-executable instructions for performing the method of claim 1.

[c14] 14. A downloadable set of processor-executable instructions for performing the method of claim 1.

[c15] 15. A database system performing recovery operations using an optimal number of recovery threads, the system comprising:
a database system having at least one database that may require recovery;
an initial recovery thread that is spawned to perform recovery operations, wherein the system measures I/O (input/output) performance with the initial recovery thread; and
a plurality of additional recovery threads that are

spawned to perform recovery operations, wherein the system measures I/O (input/output) performance with each additional recovery thread that is spawned, and wherein the system ceases spawning additional recovery threads when I/O performance degrades beyond a desired amount.

[c16] 16. The system of claim 15, wherein I/O performance is measured over a given period of time.

[c17] 17. The system of claim 16, wherein the given period of time is about 1 second.

[c18] 18. The system of claim 15, wherein the system may spawn additional recovery threads as long as I/O performance degrades by no more than about 15 percent.

[c19] 19. The system of claim 15, wherein the plurality of additional recovery threads spawned is limited such that only a maximum number of recovery threads may be generated.

[c20] 20. The system of claim 19, wherein the maximum number of recovery threads is limited to not exceed a count of databases to be opened.

[c21] 21. The system of claim 19, wherein the maximum number of recovery threads is limited to not exceed one less

than a count of database engines online.

[c22] 22. The system of claim 15, wherein, when I/O performance measured for a just-spawned recovery thread degrades beyond the desired amount, the system puts the thread to sleep.

[c23] 23. The system of claim 22, wherein the system awakens the thread that has been put to sleep, upon termination of another thread.

[c24] 24. The system of claim 15, wherein a maximum number of recovery threads permitted is limited to a configured maximum number of databases that can be recovered concurrently.

[c25] 25. The system of claim 15, wherein each recovery thread recovers a particular database at a given point in time.

[c26] 26. The system of claim 15, wherein a user of the system is able to specify a particular number of concurrent recovery threads, and wherein the system generates an advisory if the particular number of concurrent recovery threads specified is not optimal.

[c27] 27. In a database system, an auto-tuning method for performing database recovery, the method comprising:

spawning a thread to perform database recovery and recording statistics about performance associated with that thread;

attempting to spawn additional threads to perform database recovery and recording statistics about performance associated with each additional thread spawned; and

if the performance for a given thread degrades beyond a desired amount, freezing execution of the thread and ceasing any attempt to spawn additional threads for database recovery.

[c28] 28. The method of claim 27, wherein performance comprises I/O (input/output) performance measured over a given period of time.

[c29] 29. The method of claim 28, wherein the given period of time is about 1 second.

[c30] 30. The method of claim 27, wherein the desired amount is no more than about 15 percent degradation in performance.

[c31] 31. The method of claim 27, wherein only a certain maximum number of threads may be generated for performing database recovery.

[c32] 32. The method of claim 31, wherein the maximum

number of threads is limited to not exceed a count of databases that may be opened.

[c33] 33. The method of claim 31, wherein the maximum number of threads is limited to not exceed one less than a count of database engines online.

[c34] 34. The method of claim 27, wherein the freezing step comprises:
when I/O performance of the system degrades beyond a preselected percentage, freezing the thread.

[c35] 35. The method of claim 27, further comprising:
after another one of the threads finishes, thawing the thread that has been frozen.

[c36] 36. The method of claim 27, wherein the system does not attempt to spawn more threads than a maximum number of databases available to be recovered concurrently.

[c37] 37. The method of claim 27, wherein each thread recovers a particular database at a time.

[c38] 38. The method of claim 27, wherein a user of the system is able to specify a particular number of concurrent threads, and wherein the system generates an advisory if the particular number of concurrent threads specified is

not optimal.

[c39] 39. A computer-readable medium having processor-executable instructions for performing the method of claim 27.

[c40] 40. A downloadable set of processor-executable instructions for performing the method of claim 27.

[c41] 41. In a database system having a cache for database operations, a method for auto-tuning the cache to optimize database recovery operations, the method comprising:
partitioning the cache such that it includes memory allocation for both a large I/O (input/output) pool and a small I/O pool;
during normal database operation, increasing the small I/O pool's memory allocation in the cache, so that the cache is optimized for random access retrieval of data pages using memory from the small I/O pool; and
during database recovery operation, increasing the large I/O pool's memory allocation in the cache, so that the cache is optimized for sequential retrieval of log pages using memory from the large I/O pool.

[c42] 42. The method of claim 41, wherein the small I/O pool includes memory blocks with a size based on data page

size of data pages used by the database system.

[c43] 43. The method of claim 41, wherein the large I/O pool includes large memory blocks of about eight times the data page size of data pages used by the database system.

[c44] 44. The method of claim 41, further comprising:
during database recovery operation, using the large I/O pool for retrieval of log pages and the small I/O pool for retrieval of data pages.

[c45] 45. The method of claim 41, further comprising:
during database recovery operation, dynamically creating the large I/O pool if the large I/O pool is unavailable.

[c46] 46. The method of claim 41, wherein said step of increasing the large I/O pool's memory allocation includes resizing the large I/O pool for optimal recovery performance.

[c47] 47. The method of claim 41, further comprising:
during database recovery operation, resizing the small I/O pool for optimal recovery performance.

[c48] 48. The method of claim 41, further comprising:
during normal database operation, making a portion of the cache available for prefetching data pages and log

pages; and

during database recovery operation, increasing the portion of the cache available for prefetching data pages and log pages, so as to optimize the cache for database recovery operation.

[c49] 49. The method of claim 41, further comprising:

after completion of the database recovery operation, restoring the small I/O pool's memory allocation in the cache, so that the cache is optimized for random access retrieval of data pages using memory from the small I/O pool.

[c50] 50. A computer-readable medium having processor-executable instructions for performing the method of claim 41.

[c51] 51. A downloadable set of processor-executable instructions for performing the method of claim 41.

[c52] 52. In a database system having a cache for database operations, a method for optimizing the cache for retrieval of pages for database recovery operations, the method comprising:

during normal database operations, making a portion of the cache available for prefetching pages by bringing pages into the cache before the pages are used;

during database recovery operations, increasing the portion of the cache available for prefetching pages, so as to optimize the cache for retrieval of pages for database recovery operations; and

after completion of the database recovery operations, decreasing the portion of the cache available for prefetching pages so as to restore the portion of the cache available for prefetching pages for normal database operations.

[c53] 53. The method of claim 52, wherein prefetching pages includes determining pages to be brought into the cache based on established data access patterns.

[c54] 54. The method of claim 52, wherein prefetching pages includes scanning log pages to determine data pages needed for database recovery operations.

[c55] 55. The method of claim 52, wherein the portion of the cache available for prefetching pages during normal database operations is about 10 percent.

[c56] 56. The method of claim 52, wherein the portion of the cache available for prefetching pages during normal database operations is user configurable.

[c57] 57. The method of claim 52, wherein the portion of the cache available for prefetching pages during database

recovery operations is auto-tuned to about 80 percent.

[c58] 58. The method of claim 52, wherein the pages that are prefetched include data pages and log pages.

[c59] 59. The method of claim 52, further comprising:
consulting a timestamp on a data page to determine whether to prefetch the page into the cache.